(19) The Japanese Patent Office
(12) Laid-Open Patent Application Publication (A)
(11) Laid-Open Patent Application Publication No. Hei-10-31490
(43) Publication Date: February 3, 1998
(51)

Int. Cl.[5]	Classification Symbol	JPO Ref. No.	FI	
G10H 7/00			G10H 7/00	511 C
1/00	102		1/00	102 Z

Request for Examination: not yet requested
Number of Claims: 6 OL (12 pages in total)

(21) Application No. Hei-8-189480
(22) Application Date: July 18, 1996
(71) Applicant: 000116068
Roland Kabushiki Kaisha
No. 1-4-16, Dojimahama, Kita-ku, Osaka-shi, Osaka, Japan
(72) Inventor: Shigeru TAKAHASHI
c/o Roland Kabushiki Kaisha
No. 1-4-16, Dojimahama, Kita-ku, Osaka-shi, Osaka, Japan
(72) Inventor: Hiroshi KATAYAMA
c/o Roland Kabushiki Kaisha
No. 1-4-16, Dojimahama, Kita-ku, Osaka-shi, Osaka, Japan
(74) Agent: Patent Attorney, Takao KOBAYASHI (and other 1 person)

(54) [Title of the Invention] SOUND MATERIAL PROCESSOR FOR AN ELECTRONIC MUSICAL INSTRUMENT

(57) [Abstract]
[Problem to be Solved] The present invention relates to a sound material processor for an electronic musical instrument for dividing a sound material, created in a live performance by a player, into one sound. An object of the invention is to divide

the sound material, created in the live performance, into blocks of sounds appropriate to the beat speed in the performance. [Constitution] A sound material processor comprises: a first storage means for storing a parent wave formed by sampling a sound material made up of a series of sounds; a dividing means for dividing the parent wave read out by the first storage means into plural child waves, using a threshold to detect each sound rise; and a second storage means for storing the results of the division executed by the dividing means.

[Detailed Description of the Invention]

[0001]

[Technical Field of the Invention]

The present invention relates to a sound material processor for an electronic musical instrument for dividing a sound material, created in a live performance by a player, into one sound.

[0002]

[Prior Art]

There is a technique available for playing a synchronized performance of sound materials previously recorded in a live performance, such as rhythm musical instrument sounds, voices and various sound effects, with other sounds created in a different performance. For example, JP-A-H07-244480 discloses a performance data processing method for an electronic musical instrument, which is adapted: to record a series of performance sounds, created in a live performance by a player, as PCM data; to divide the recorded performance data on the time axis by equal volume into blocks of plural data; to assign a key number (note number) to each block of performance data; and to read and play the block of performance data corresponding to the key number of a key on a keyboard or the like pressed by a player with the timing at which another player generates a performance sound, so that multi-sound recording synchronized with other sounds created in a different performance can be easily implemented.

[0003]

[Problem to be Solved by the Invention]

However, the aforementioned performance data processing method is adapted to divide the performance data by equal volume into blocks, which may cause the divided blocks not to match the beat. The following chart shows aspects when the performance in one 16- or 4-beat bar is divided into equal volume.

[0004]

[Chart: Division of one-bar performance into equal volume]

Number by which performance is divided	4	16
16-beat performance	4 performance sounds per block	1 performance sound per block
4-beat performance	1 performance sound per block	1/4 performance sound per block

[0005]

For example, as shown in FIG. 11(a), 16-beat performance data is divided into four blocks, and each block consists of four performance sounds. When the 16-beat performance data divided into four blocks is synchronized with other performance sounds whose performance speed is half of this 16-beat performance data, each time the player presses a key with the timing of another performance sound, four performance sounds in one block assigned with a key number corresponding to the pressed key are played at the same tempo as the original performance data, as shown in FIG. 11(b). This results in a blank between sounds produced by the current and next key presses, which generates unnatural performance sounds.

[0006]

If the 16-beat performance data is divided into 16 blocks, that is, each block has a single performance sound, then this single performance sound is played per key press with the timing of another performance sound, resulting in no problem.

[0007]

In addition, as shown in FIG. 11(c), 4-beat performance data is divided into four blocks, and each block consists of a single performance sound. This results in no problem at all. However, as shown in FIG. 11(d), if the four-beat performance data is divided into 16 blocks, that is, each block has a one-fourth of a single performance sound wave, reproduction thereof sounds unnatural:

[0008]

As a countermeasure against these problems, the aforementioned performance data processing method may be adapted to select the number of blocks to be divided as required. Nevertheless, this method is not applicable to the case where only part of the performance is played with a faster beat.

[0009]

Therefore, an object of the present invention it to divide a sound material, created in a live performance, into blocks of sounds appropriate to the beat speed in the performance. Another object of the invention is to immediately and automatically perform individual data resulted from the division.

[0010]

[Means for Solving the Problem]

In order to solve the aforementioned problems, a sound material processor for an electronic musical instrument of the present invention comprises: a first storage means for storing a parent wave formed by sampling a sound material made up of a series of sounds; a dividing means for dividing the parent wave read out by the first storage means into plural child waves, using a threshold to detect each sound rise; and a second storage means for storing the results of the division executed by the dividing means. Using the threshold in such a manner allows the sound material to be divided into blocks appropriate to the beat speed in the performance.

[0011]

The aforementioned dividing means may be adapted to include

another means for redividing one of the divided child waves, which still has at least two sounds, into additional plural child waves, using a threshold different from the threshold used for the first division to redetect sound rises of the child wave to be redivided.

[0012]

Alternatively, the aforementioned dividing means may be adapted to include another means for redividing one of the divided child waves, which still has at least two sounds, into additional plural child waves, by specifying a temporal position on the time axis of the child wave to be redivided.

[0013]

The sound material processor for an electronic musical instrument according to the invention further has: a designation means for designating each of the divided child waves; and a reproduction means for reproducing the child wave designated by the designation means. Sequentially designating each child wave by such designation means can result in reproduction of the original parent wave.

[0014]

The sound material processor for an electronic musical instrument according to the invention further has: a sequence data creating means for creating sequence data including information about the timing at which the child waves sound in series based on the child wave data resulted from the division and stored in the second storage means. Based on this sequence data, the individual divided child waves can be immediately and automatically played, and the original parent wave can therefore be reproduced.

[0015]

As well as a means for reproducing the sequence data, the sound material processor for an electronic musical instrument according to the invention further has: a reproduction means for reproducing each child wave based on the sequence data; and a synchronization means for synchronized play with another performance data by matching the timing information of the

sequence data with timing information of another performance data at the time of reproduction by the reproduction means. Using such synchronization means facilitates synchronized play with another performance data.

[0016]

[Embodiment of the Invention]

An embodiment of the present invention is described with reference to the accompanying drawings as follows. FIG. 2 is a block diagram, showing an electronic musical instrument as an embodiment of the present invention. The electronic musical instrument has a function to divide performance data and synchronize the divided performance data with tempo information. The electronic musical instrument system includes: a central processing unit (CPU) 1 for executing performance data dividing process and the entire control; a read only memory (ROM) 2 for storing a program and the like for controlling the entire system; a random access memory (RAM) 3 for storing the performance data and being used as a working area for the CPU; a display section 4 and an operator section 5 for checking and operating a control parameter; an A/D converter section 6 for inputting the performance data; a D/A converter section 7 for outputting the performance data; a tone cooler controller section 8 for controlling a tone color of the output performance data and the like; and a MIDI input/output terminal 9 for inputting/outputting the performance data and MIDI clock data.

[0017]

FIG. 3 shows an example of an operator panel for operating the electronic musical instrument. The operator panel is made up of the display section 4 for displaying the control parameter and the like, and the operator section 5 for implementing various operations such as setting the parameter. Various operators 51 to 55 form the operator section 5. Reference numeral 51 denotes a dial type of rotary encoder for setting the parameter by changing parameter values displayed on the display section 4. The parameters set by the rotary encoder 51 include threshold level, start point, end point, pitch, and

level. Reference numerals 52 to 54 denote a switch for selecting between YES/NO commands, an analyze switch for giving the command to execute the performance data dividing process; and a parameter selection switch for selecting a parameter to be displayed on the display section 4, respectively. Reference numerals 55_1 to 55_4 denote reproduction switches for allowing the divided performance data to sound for checking. Indeed, the number of reproduction switches to be prepared is equivalent to the number of divided child waves, although the drawing only shows an example with four reproduction switches.

[0018]

A series of performance data, which has not yet been divided, is referred to as parent wave herein, while each of the individually divided performance data is referred to as child wave. The reproduction switches 55_1 to 55_4 are adapted to respectively correspond to the plural divided child waves, as will be discussed later. Pressing either one of the reproduction switches 55_1 to 55_4 allows its corresponding divided child wave to be reproduced. In the case where there are several parent waves to be covered by the dividing process, the reproduction switches 55_1 to 55_4 are also adapted to respectively correspond to these parent waves. Pressing either one of the reproduction switches 55_1 to 55_4 concurrently with pressing the analyze switch 53 allows entering the performance data restructuring mode for the parent wave corresponding to the pressed reproduction switch.

[0019]

Operation of the electronic musical instrument of this embodiment is described. The principle operation is initially described with reference to FIG. 1.

[0020]

FIG. 1 shows an example in which certain performance data is divided, in which FIG. 1(a) shows a waveform of an original parent wave and FIG. 1(b) shows an aspect of the dividing process. Bar lines forming the waveform in the drawing indicate their respective sample values of a sampled sound waveform from the

performance. As shown in FIG. 1(b), a first threshold level TH1 is set for the parent wave, and one division block is defined by portion that the parent wave exceeds the first threshold level TH1, then decreases to TH1 or lower, and then exceeds TH1 again. This results in seven divided child waves.

[0021]

In such a case, the first, second, third, fourth, sixth and seventh child waves individually have a single peak value for the performance sound per block. In other words, these child waves individually include its single performance sound. In contrast, the fifth child wave has two peak values per block, that is, it includes two performance sounds. This requires the fifth child wave to be further divided. Thus, the fifth child wave is designated and another appropriate threshold level TH2, which is greater than the first threshold level TH1, is reset to retry the second division. As shown by the example of FIG. 1(b), the second division allows the fifth child wave resulted from the first division to be further divided into two. This results in total eight divided child waves after the second division.

[0022]

For the purpose of preventing a peak value, allegedly derived from the noise, from defining a block, a minimum time period per block is preset at e.g. 100ms, and in the event that a block is to be defined shorter than the preset minimum time period, it may be automatically incorporated into the following block.

[0023]

The following describes a process for executing the principle operation. FIG. 4 is a flowchart of a routine for restructuring the performance data. In a panel process routine, designed to scan the panel switches to check the status thereof, of the main routine (not shown), the routine for restructuring the performance data is read out, taking advantage of detecting which one of the reproduction switches 55_1 to 55_4 is pressed with the analyze switch 53 being pressed. The parent wave, which corresponds to the pressed reproduction switch, undergoes

the performance data restructuring process.

[0024]

When the process enters the performance data restructuring mode, it checks whether or not the applicable parent wave has already been divided (step S2). If the parent wave has already been divided, the process selects, by pressing a specific switch, either a play mode for reproducing the divided performance data or an analyze mode for redividing the performance data (step S3). If selecting the play mode, the process completes the routine for restructuring the performance data (step S10). If the process determines that the parent wave has not yet been divided in the step S2, it enters the analyze mode.

[0025]

In the analyze mode, the process searches for a maximum and minimum value of the sample values forming the applicable parent wave (step S4), defines a value, obtained by subtracting the minimum value from the maximum value and dividing by 128, as the scale of resolution, and presets a threshold (sense value) using the operator based on the resolution (step S5). An operating person may manually preset the threshold by visually adjusting the value displayed on the display section 4. Alternatively, the control program may automatically preset a value specific to the first and second dividing process. As for presetting the threshold, delaying the display of the waveform of absolute values for the parent wave helps a user easily preset the threshold.

[0026]

After the threshold is completely preset, an inquiry about whether or not the division is executed is displayed on the display section 4. If the division is to be performed, an YES switch 52 is pressed. If it is not to be executed, a NO switch 52 is pressed (step S6). The dividing process is executed by pressing the YES switch 52 (step S7). The routine for restructuring the performance data is completed by pressing the NO switch 52 (step S10). Details of the dividing process (step S7) will be described later with reference to the flowchart

shown in FIG. 5.

[0027]

After the dividing process is completed, sequence data is created based on the divided performance data and the timing for producing sounds. A method for creating the sequence data will be discussed later.

[0028]

After the dividing process is completed, an inquiry about whether or not the dividing process is re-performed is displayed on the display section 4 (step S9). Since the sequence data, for playing the individual performance data (child waves) divided through the last dividing process, are assigned to the reproduction switches 55_1 to 55_4, operating the reproduction switches 55_1 to 55_4 allows the corresponding individual divided performance data to be reproduced for checking. Reproducing and checking these performance data in such a manner permits a user to audibly determine whether or not the parent wave is properly divided. Thus, if it is properly divided, the "NO" command, that is, no more execution, is given by pressing the NO switch 52. If the parent wave is not properly divided, the "YES" command, that is, re-execution, is given by pressing the YES switch 52. One of the options can be selected at this moment through operating the operator section 5 while viewing the display on the display section 4. One option is ALL selection, more specifically, re-executing all the steps of the parent wave dividing process from the beginning by resetting the threshold. The other option is partial re-execution of the dividing process for only part of the child waves, which could not be properly divided, by resetting the threshold.

[0029]

When the "YES" command for re-execution is given, the process repeats the steps S5 to S8. Only part of the child waves are divided by resetting the threshold, which means this is at least the second time the dividing process is executed. Repeating the dividing process in such a manner allows the parent wave to be divided into child waves such that each child wave can

eventually consist of only a single performance sound.

[0030]

The performance data dividing process is described with reference to the flowchart shown in FIG. 5. A pointer PTR is set at the beginning of the performance data for the parent wave to be divided (step S702). The pointer PTR is then incremented by one sequentially, which allows a point at which the wave undergoes the dividing process to be indicated by the number of samples. When the command of re-execution is given in the step 9 of the flowchart of FIG. 4, if the ALL selection (re-execution of all the steps of the dividing process) is taken, then the pointer PTR is set at the beginning of the parent wave, or if a certain child wave is selected, the pointer PTR is set at the beginning of the certain child wave.

[0031]

Next, a start point (the beginning point of the block for the first child wave) is preset as a parameter for the child wave. In other words, the start point is written into a parameter buffer for storing the start point for the child wave (step S703). Also in this case, if the ALL selection is taken in the step S9 for re-execution, then the start point, set at the beginning of the parent wave, is written into the parameter buffer for the first child wave, or if a certain child wave is selected, the start point that has already been written into the parameter buffer for the certain child wave is used without change.

[0032]

In addition to that, an after-mentioned end point (a rearmost point of the block for the child wave) is also written into the parameter buffer for this certain child wave.

[0033]

Then, a FLAG is turned OFF (step S704). The FLAG is designed to identify a beginning of the next child wave and be turned ON by the fact that a sample value of the child wave is smaller than the threshold (sample value < threshold). Then, if the FLAG, remaining ON, detects a sample value greater than the threshold (sample value = threshold), this indicates a

beginning of the next child wave.

[0034]

In the above description, the FLAG is inverted when the sample value is smaller than the threshold. To be more specific, in order to prevent the FLAG from being incorrectly inverted when the sample value is in the vicinity of a zero-crossing point, the FLAG is adapted to be ON for the first time after a predetermined times of confirmation that the sample value is actually smaller than the threshold. Alternatively, a maximum value of absolute values per 10 ms, for example, is sequentially compared with the threshold, as another embodiment. Each amplitude shown in FIG. 1 or FIGs. 10(a) to (d) indicates the maximum value of the absolute values per 10ms. A dividing point is determined based on this maximum value.

[0035]

After that, the amplitude level (sample value) of the performance data is found per sample using the pointer PTR to determine whether or not the sample value is greater than the threshold (step S705). If the sample value is greater than the threshold, that is, the performance sound is detected (step S705), then a determination is made whether or not the FLAG is ON (step S706). If the FLAG is OFF, the pointer PTR is incremented by one (step S707) to compare the threshold with the next sample value (step S705).

[0036]

If the sample value is lower than the threshold, that is, a determination is made that the detected performance sound was completely played, the pointer PTR is incremented by one while the FLAG is turned ON (step S708) to compare the threshold with the next sample value (step S705).

[0037]

The FLAG is determined to be ON in the step S706, which means that another performance sound is detected after the determination that the performance sound was completely played. In other words, the beginning of the block for the next child wave is detected. In this case, a determination is made whether

or not a length of the child wave reaches a certain length (step S709). If it does not reach the certain length, then there is a higher possibility that what was supposed to be a detected performance sound is allegedly noise. Thus, the FLAG is turned OFF again (step S710), and the pointer PTR is incremented by one (step S707) to compare the threshold with the next sample value again (step S705). The reason for making the determination if the length of the child wave is equal to or greater than the certain length in the step S709 is because of removing noise, as described above. Renumbering the child waves to be consecutive is also required.

[0038]

If the length of the child wave reaches the certain length (step S709), an end point of the child wave is written into the parameter buffer for storing the end point, and the FLAG is set to OFF (step S711). This allows certain points on the parent wave to be written as its start and end points into the parameter buffer for the child wave as the results of the dividing process. This defines a point, at which the FLAG is turned OFF, as a division point. To be more specific, a point, which first turns positive or crosses zero immediately before the division point thus defined, is defined as a start address and therefore an address before the start address is defined as an end address.

[0039]

In this embodiment, the end and start addresses are detected at the above point in time. Alternatively, a next zero-crossing point, which is obtained immediately after a predetermined number of times that a sample value is lower than the threshold level, may be defined as the end address of the child wave. The start address of the next child wave may be a zero-crossing address, which is obtained immediately before the sample value exceeds the threshold level, as described above. Further, threshold levels for detecting the end address and the start address may be separately preset.

[0040]

Next, the pointer PTR is incremented by one (step S712) to

refer to the next sample value, and a determination is made whether or not the next sample value reaches the end of the wave to be divided (step S713). If it does not reach the end of the wave, a start point for the next child wave is written into a parameter buffer for storing such start point (step S714) to repeat the process from the step S705 onward. If it reaches the end of the wave, the routine for the dividing process is completed (step S715).

[0041]

Now, a method for creating sequence data for the parent wave based on the plural child waves resulted from the above dividing process is described with reference to FIGs. 7 and 8. As shown in FIG. 7, based on the plural child waves resulted from the above dividing process and their parameters, a wave number is given to each child wave in order from the first child wave. Concurrently, with respect to the start point for each divided child wave, a gate time is calculated, according to the timing at which the child wave sounds, to create the sequence data. The gate time refers to an interval between the sounds of a child wave and the next child wave. The sequence data is constituted as [gate time + child wave number]. After the lapse of the gate time, the sound of the child wave is reproduced from its start to end points. The data of the gate time and child wave number is respectively one byte.

[0042]

The procedure for creating the sequence data includes the following steps.

(1) Assuming the start point for the first child wave to be reproduced (the wave number of 1 in the chart) as the first gate time.

(2) Assuming a value, obtained by subtracting the start point for the first child wave to be reproduced from the start point for the second child wave to be reproduced, as the second gate time.

(3) Assuming a value, obtained by subtracting the start point for the N^{th} child wave to be reproduced from the start point

for the $N+1^{th}$ child wave to be reproduced, as the N^{th} gate time.
(4) Assuming the last wave number as "FF." Completing reproduction at the time of reproducing the sound of the child wave with the number "FF."

[0043]

FIG. 7 shows an example of the parameters for the child waves for creating the sequence data. FIG. 8 shows an example of the sequence data created. A hexadecimal digit is used for displaying the start/end points and gate time. In the event that the gate time exceeds the length of "FF," the next one byte is also used for this gate time. FIG. 8 shows an example in which the fourth gate time has a length of "FF + 13." As the performance tempo is changed, a value of the gate time is changed, accordingly.

[0044]

The aforementioned child wave numbers, that is, the individual divided child waves, are assigned to the corresponding reproduction switches 55_1 to 55_4.

[0045]

Alternatively, a note number (key number) may be assigned to each child wave number. FIG. 9 shows an example for that. This example shows four types of parent waves, A, B, C and D, each of which is divided into 16 child waves, such as A-1 to A16, B-1 to B16, C1 to C16, and D1 to D16, respectively. The corresponding note numbers are automatically assigned to the child waves in order, beginning with the child wave A-1, as shown in FIG. 9. This results in creation of the sequence data consisting of [gate time + note number]. Arranging the note numbers in the sequence data in the order of the scale allows clear reproduction of the original performance sounds of the original parent wave based on the sequence data.

[0046]

The aforementioned example for the sequence data shows that the timing information included in the sequence data is represented by the gate time, which varies depending on the performance tempo. Other than that, the timing information may

be represented using MIDI clock, so that the timing information may be matched with the performance tempo by external MIDI clock.

[0047]

For example, where Fs [Hz] is a sampling frequency of the parent wave, a required time period per sample is represented as 1 sample = 1/Fs [sec], in which tempo for the parent wave is defined as T[bpm]. This scale [bpm] indicates the number of quarter notes per minute. Since a single quarter note corresponds to 24 MIDI clock, a length of MIDI clock, T_{CLK}, is represented as $T_{clk} = T/60 \times 24$ [sec]. Thus, a time interval between sounds produced from the first and second child waves is represented as $(N_2 - N_1)/F_S$ [sec] = $[(N_2 - N_1)/F_S] / [T/(60 \times 24)]$ (scale [MIDI clock]), where N_1 is start point for the first child wave, and N_2 is start point for the second child wave.

[0048]

In such a manner described above, the timing information, included in the sequence data, is represented using MIDI clock. This makes it easier to synchronize sounds of the parent wave reproduced based on the sequence data with other reproduced sounds of another performance data, by controlling the timing information included in the sequence data, using internal or external MIDI clock.

[0049]

Now, description is made for reproduction process for the sequence data created. FIG. 6 is a flowchart of a routine for reproducing the sequence data. The reproduction routine is a panel processing routine included in the main routine (not shown), which is read out and executed by pressing either one of the reproduction switches 55_1 to 55_4 (step S11).

[0050]

The first one byte of the sequence data, that is gate time, is stored in a counter area (step S12). If the value stored indicates "FF," the FLAG is turned ON. This value stored in the counter area serves as a counter since the value is sequentially decremented per predetermined time period. A

pointer is placed at the beginning of the sequence data. The pointer is designed to manage the progress of reproducing the sequence data by byte.

[0051]

Next, a timer starts-up (step S13). A timer interrupt occurs every time the timer shows a lapse of a predetermined time period. The process determines whether or not a timer interrupt occurs (step S14). If the process receives the timer interrupt, the value preset in the counter area (gate time) is decremented by one (step S15). The process determines whether or not the counter value is equal to or lower than 1, and if the counter value is other than "0," it continues to be decremented per timer interrupt (steps S14 to S16).

[0052]

If the counter value becomes "0" (step S17), which means that the preset gate time has elapsed, the pointer is incremented by one (step S17). With reference to the FLAG, a determination is made whether or not the data (gate time) preset in the counter area indicates "FF" (step S18). If the data indicates "FF" (step S714), the gate time should be "FF + oo," as previously described for creating the sequence data. This means that the gate time has not elapsed yet while the counter value continues to be decremented. Thus, data, one byte next to the data "FF," that is, the remaining gate time oo, is read out of the sequence data and is stored in the counter (step S19). The counter value is repeatedly decrement until it becomes "0" (steps S14 to S19).

[0053]

In the determination in the step S18, unless the next data indicates "FF," the FLAG is reset to OFF and the child wave number is stored in a work area to set the data for the next gate time, read out of the sequence data, on the counter (step S20).

[0054]

After that, the child wave reproduction process is executed (step S21). After the child wave reproduction process is completed, a determination is made whether or not the sequence data is ended (step S22). The process from the step S14 to the

step S22 is repeated until the sequence data is ended. If the sequence data is ended, the routine for reproducing the sequence data is completed (step S23).

[0055]

For reproducing the child waves, tone color control may be performed per divided performance sound by the tone color controller section 8 which controls a tone color for every divided child wave read out. The tone color control may be adapted to control the tone color for the child wave read out in accordance with the information about the tone color control externally inputted. In addition, the tone color controller section may control the tone color and the like for every divided performance data, using information about controlling the tone color and the like recorded on the sequence data created.

[0056]

In the aforementioned embodiment, when the tempo becomes faster, which results in blended sounds produced by two consecutive child waves, the sound for the later one of those child waves precedes the sound for the earlier one by deadening such earlier sound or cross-fading the blended sounds.

[0057]

Various embodiments may be employed to carry out the invention. As one example shown in the aforementioned embodiment, if the first dividing process results in one of child waves having at least two performance sounds, the second dividing process is executed with a different threshold to further divide such child wave. However, the present invention is not limited to that. An operating person may listen to the reproduced child wave having at least two performance sounds to know which temporal position of the child wave each performance sound is at, and specify a point, at which the second division is performed, on the time axis of the child wave with time as a parameter through manual operation, thereby further dividing the child wave.

[0058]

In addition, in the aforementioned embodiment, a block, which has more than one performance sound resulted from the first

division, is redivided at least once with a different threshold, thereby further dividing the block so as to only have a single performance sound. Alternatively, plural thresholds are initially preset to compare the parent wave with each of the plural thresholds and to save all the comparison results. At the time of completion of this comparison process, only a single time of division is performed based on those comparison results, such that all the blocks individually have only a single performance sound.

[0059]

The aforementioned embodiment also describes, as a method for synchronizing the sequence data for the parent wave with another performance data, a change in gate time as well as a MIDI clock control by representing the timing information using MIDI clock. However, the present invention is not limited to that. For example, while the sequence data remains unchanged, reproduction timing may be relatively varied. To be more specific, in the routine for reproducing the sequence data shown in FIG. 6, the timer, adapted to start in the step S13, may be designed to preset a certain value of its timer counter TC, which is decremented by one per given time interval. Then, when the TC value is equal to 0, the timer interrupt occurs. This allows the timer interrupt to occur more frequently with a TC value preset smaller, while allowing the timer interrupt to occur at a longer interval with a TC value preset greater. Thus, in order to change the reproduction timing internally, a certain TC value may be preset internally. In turn, in order to change the reproduction timing externally, such a certain TC value may be preset using an external clock.

[0060]

In addition, in the aforementioned embodiment, in order to reproduce the divided performance data (child wave), a portion corresponding to such child wave is read out of the original performance data (parent wave). Alternatively, the performance data associated with a section between the start and end points resulted from the division may be separately

written into a memory as performance data for each child wave.

[0061]

[Effect of the Invention]

As described above, the present invention allows a sound material, created in a live performance, to be divided into blocks appropriate to the beat speed in the performance. Using the sequence data resulted from the division, the present invention also allows the performance data for the individual divided child waves to be immediately and automatically played.

[Brief Description of the Drawings]

FIG. 1 is an explanatory view for principle operation of an embodiment of the present invention.

FIG. 2 is a block diagram of a processor according to the embodiment of the present invention.

FIG. 3 shows an example of an operation panel of the processor according to the embodiment of the present invention.

FIG. 4 is a flowchart of a routine for restructuring performance data to be executed by the processor according to the embodiment of the present invention.

FIG. 5 is a flowchart of a routine for performance data dividing process to be executed by the processor according to the embodiment of the present invention.

FIG. 6 is a flowchart of a routine for sequence data reproduction to be executed by the processor according to the embodiment of the present invention.

FIG. 7 is a chart showing an example of child wave parameters for creating the sequence data for the processor according to the embodiment of the present invention.

FIG. 8 shows an example of the sequence data for the processor according to the embodiment of the present invention.

FIG. 9 is a chart, showing an example in which a note number is assigned to each child wave number.

FIGs. 10(a) to 10(d) show a division into blocks to describe a problem of difficulty in matching the divided blocks with different beat speeds in the performance to each other.

[Description of Reference Numerals]

1: CPU 2: ROM

3: RAM 4: display section

5: operator section 6: A/D converter section

7: D/A converter section 8: tone color controller section

9: MIDI input/output section

51: rotary encoder 52: YES/NO switch

53: analyze switch 54: parameter selection switch

55_1 - 55_4: reproduction switch

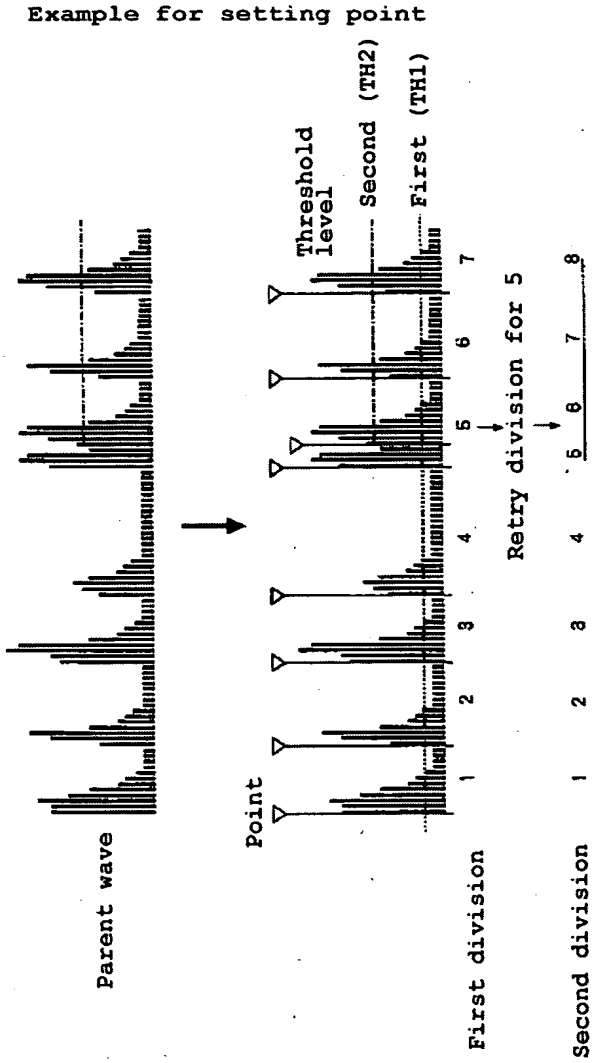

Example for setting point
Parent wave
Point
Threshold level
Second (TH2)
First (TH1)
Retry division for 5
First division
Second division
1
2
3
4
5
6
7
8

FIG. 1

Example of block diagram according to embodiment

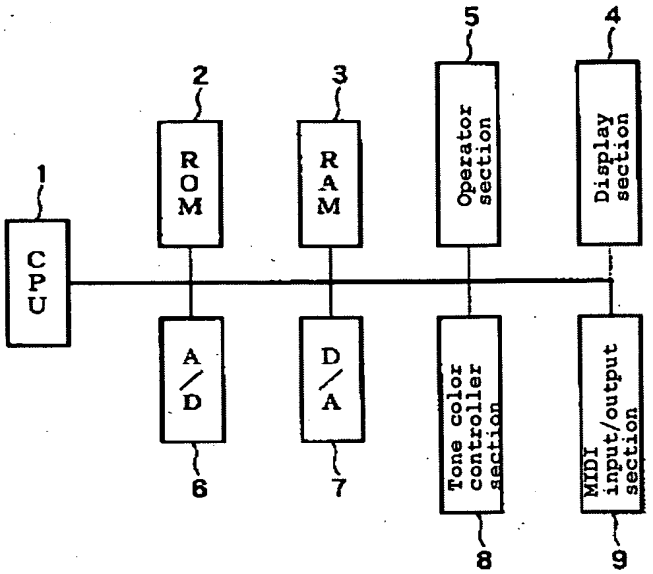


FIG. 2

Example of operation panel

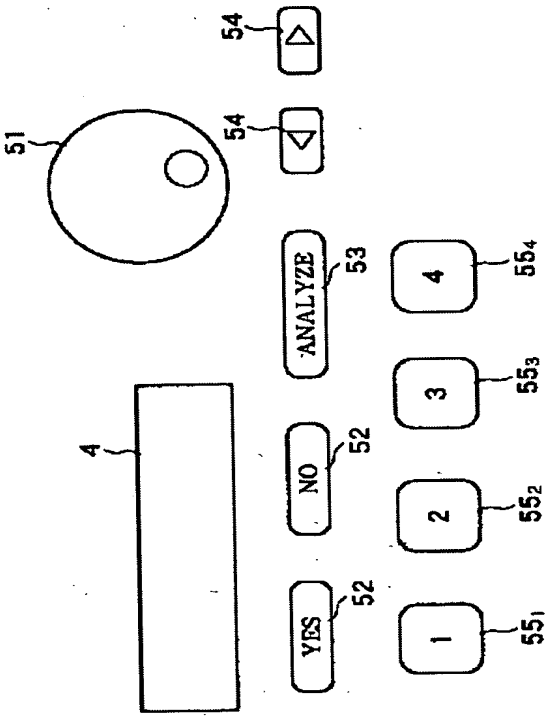


FIG. 3

Example of operation panel

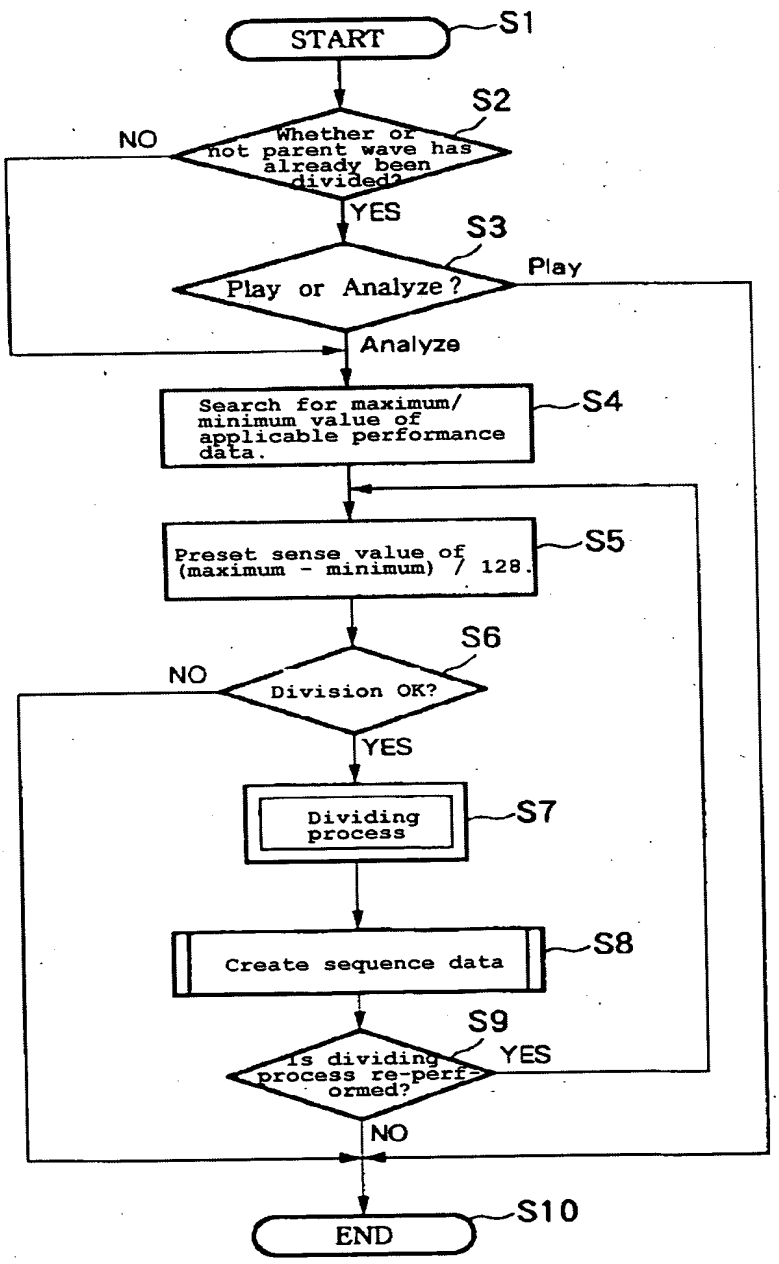


FIG. 4

Performance data dividing process flow

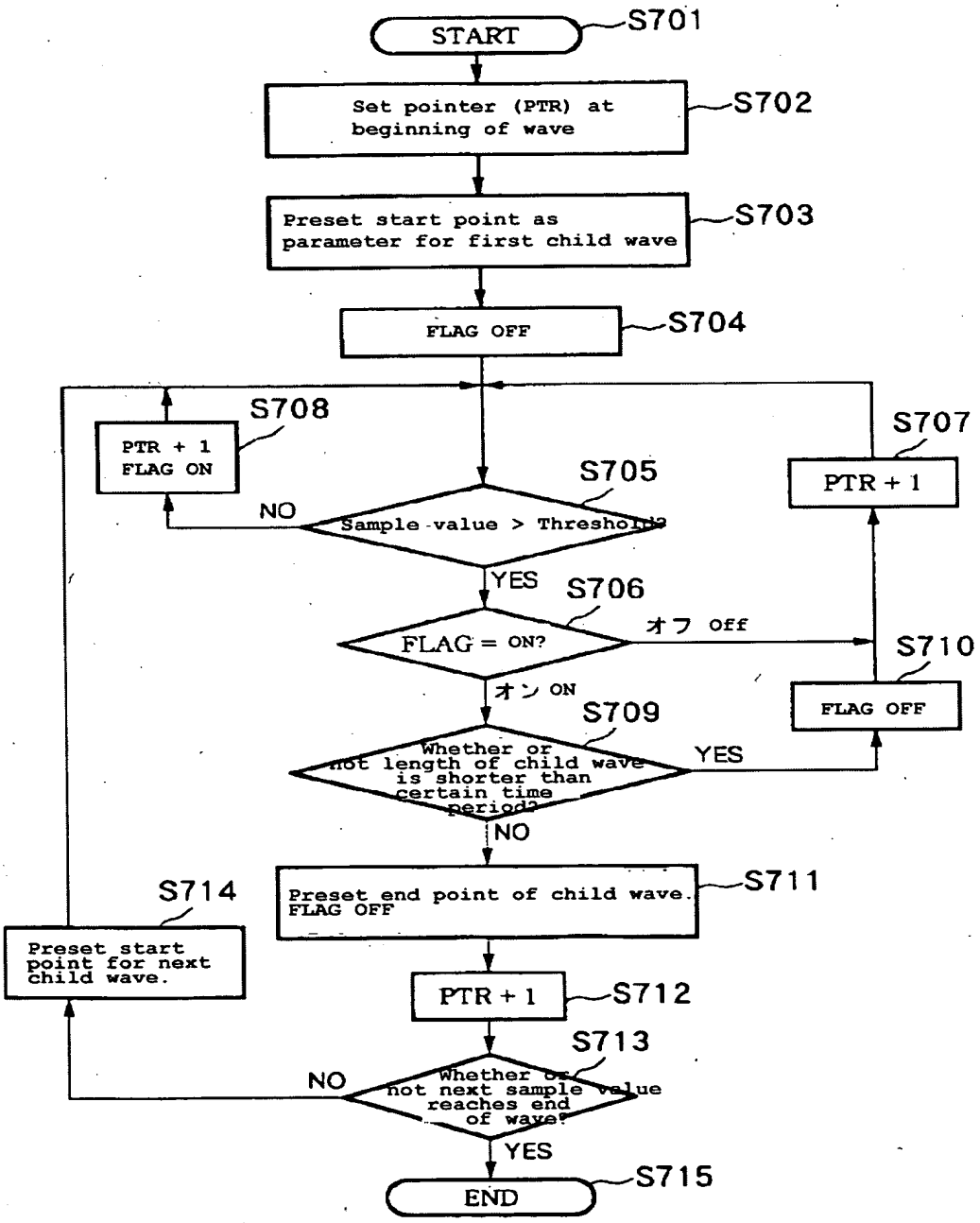


FIG. 5

Flowchart for reproducing sequence data

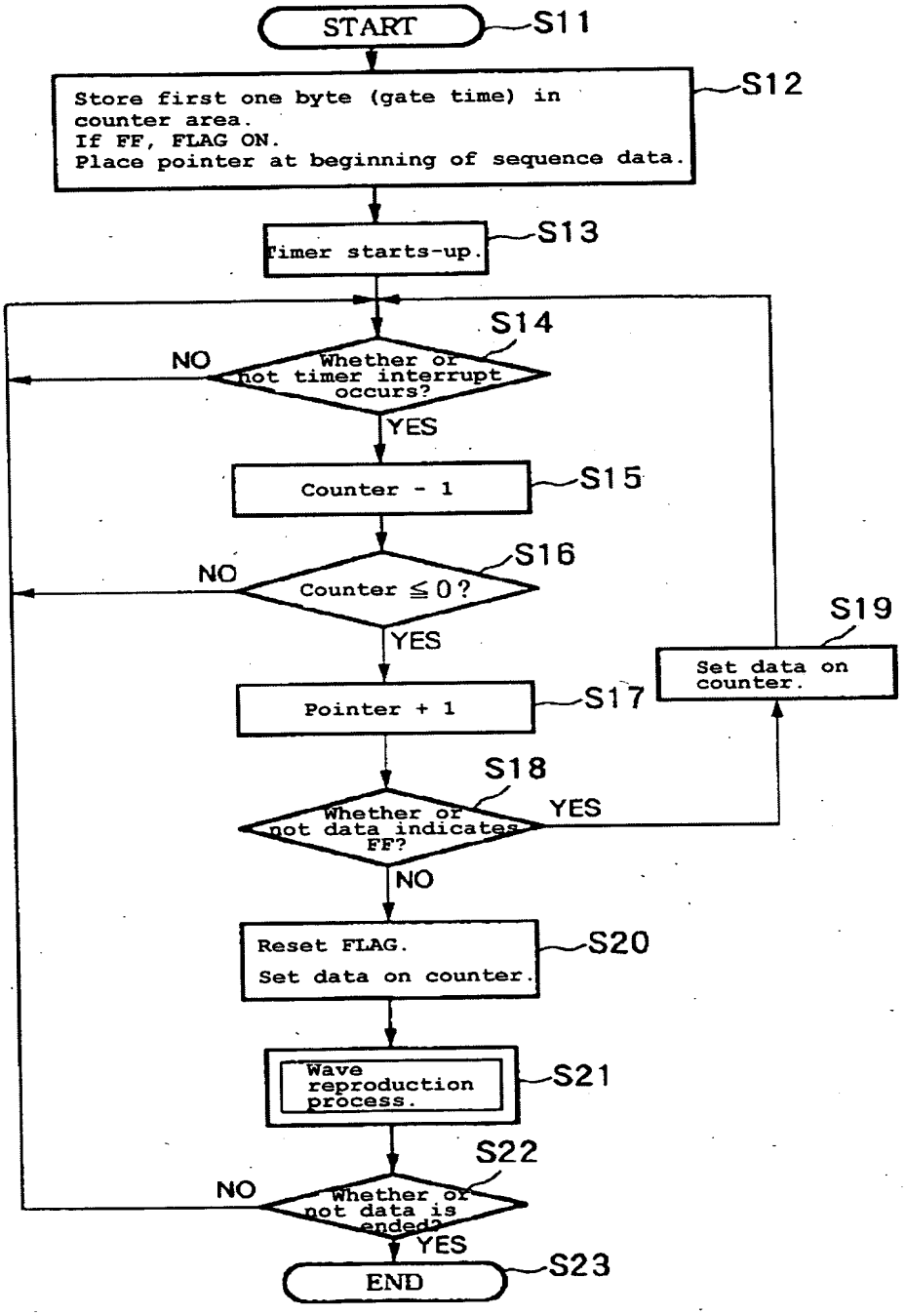


FIG.6

Example for creating sequence data

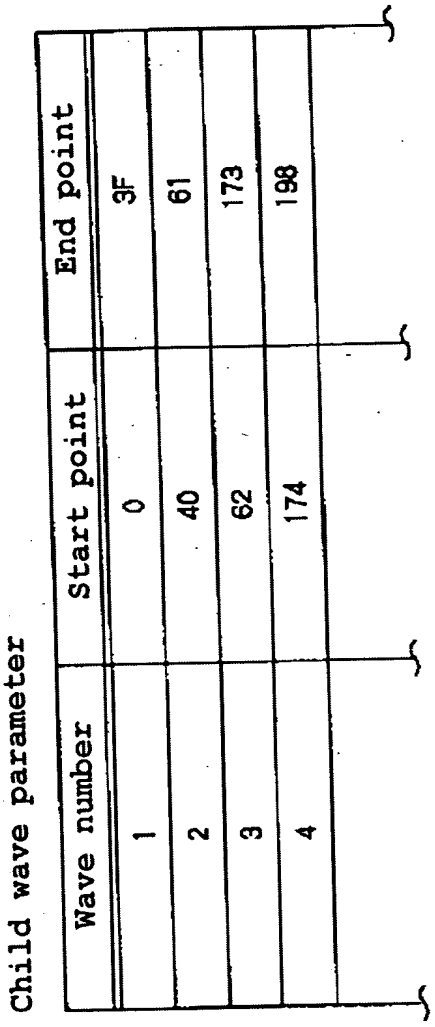


FIG. 7

Example of sequence data

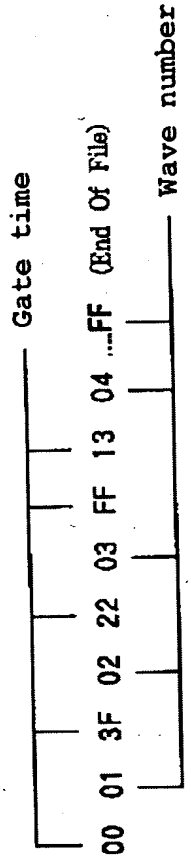


FIG. 8

Example for assigning note number to individual divided wave.

Pad note number corresponding chart (provisional)																
A-1 C1	A-2 C#1	A-3 D1	A-4 D#1	A-5 E1	A-6 F1	A-7 F#1	A-8 G1	A-9 G#1	A-10 A1	A-11 A#1	A-12 B1	A-13 C2	A-14 C#2	A-15 D2	A-16 D#2
B-1 C3	B-2 C#3	B-3 D3	B-4 D#3	B-5 E3	B-6 F3	B-7 F#3	B-8 G3	B-9 G#3	B-10 A3	B-11 A#3	B-12 B3	B-13 C4	B-14 C#4	B-15 D4	B-16 D#4
C-1 C5	C-2 C#5	C-3 D5	C-4 D#5	C-5 E5	C-6 F5	C-7 F#5	C-8 G5	C-9 G#5	C-10 A5	C-11 A#5	C-12 B5	C-13 C6	C-14 C#6	C-15 D6	C-16 D#6
D-1 C7	D-1 C#7	D-3 D7	D-4 D#7	D-5 E7	D-6 F7	D-7 F#7	D-8 G7	D-9 G#7	D-10 A7	D-11 A#7	D-12 B7	D-13 C8	D-14 C#8	D-15 D8	D-16 D#8

FIG. 9

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－31490

(43)公開日　平成10年(1998)２月３日

(51)Int.Cl.[6]	識別記号	ＦＩ		
G10H 7/00		G10H 7/00	511	C
1/00	102	1/00	102	Z

審査請求　未請求　請求項の数６　ＯＬ　（全12頁）

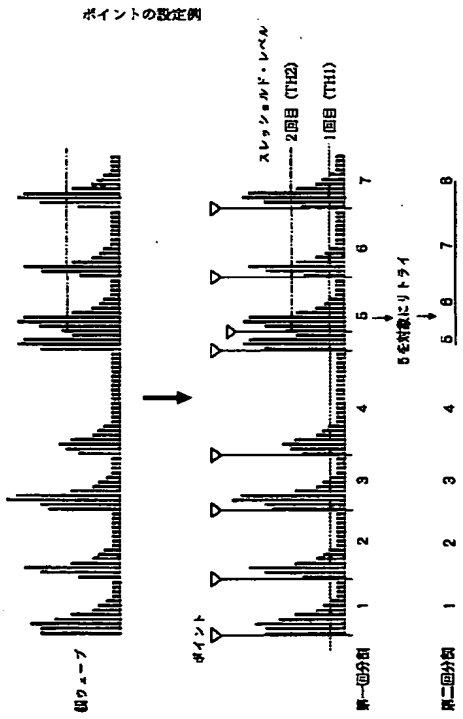
(21)出願番号　特願平8－189480

(22)出願日　平成８年(1996)７月18日

(71)出願人　000116068
ローランド株式会社
大阪府大阪市北区堂島浜１丁目４番16号
(72)発明者　高橋　茂
大阪府大阪市北区堂島浜１丁目４番16号
ローランド株式会社内
(72)発明者　片山　弘
大阪府大阪市北区堂島浜１丁目４番16号
ローランド株式会社内
(74)代理人　弁理士　小林　隆夫　（外１名）

(54)【発明の名称】電子楽器の音素材処理装置

(57)【要約】
【課題】本発明は例えば演奏者が行った生演奏などの音素材を一音一音に分割する電子楽器の音素材処理装置に関するものであり、生演奏などから得た音素材を、演奏のビートの細かさにも対応して音単位に的確に分割しブロック化できるようにすることを目的とする。
【構成】複数の一連の音からなる音素材をサンプリングした親波形を記憶する第１の記憶手段と、該第１の記憶手段から読み出した親波形を閾値を用いて該音の立上りを検出して複数の子波形に分割する分割手段と、該分割手段による分割結果を記憶する第２の記憶手段とを備える。



1

【特許請求の範囲】

【請求項１】複数の一連の音からなる音素材をサンプリングした親波形を記憶する第１の記憶手段と、

該第１の記憶手段から読み出した親波形を閾値を用いて該音の立上りを検出して複数の子波形に分割する分割手段と、

該分割手段による分割結果を記憶する第２の記憶手段とを備えた電子楽器の音素材処理装置。

【請求項２】該分割手段は、分割済の１つの子波形中に２以上の音が含まれているときに、該子波形について、該子波形を分割するに用いた閾値と異なる閾値を用いて該音の立上りを更に検出し直して該子波形を更に複数の子波形に分割する手段を含む請求項１記載の電子楽器の音素材処理装置。

【請求項３】該分割手段は、分割済の１つの子波形中に２以上の音が含まれているときに、該子波形について、該子波形の時間軸上で時間位置を指定して該子波形を更に複数の子波形に分割する手段を含む請求項１記載の電子楽器の音素材処理装置。

【請求項４】該分割した各子波形を指定する指定手段と、該指定手段で指定された子波形を再生する再生手段を更に備えた請求項１～３の何れかに記載の電子楽器の音素材処理装置。

【請求項５】該第２の記憶手段に記憶された分割結果の子波形データに基づいて各子波形を一連に発音するタイミング情報を含んだシーケンスデータを作成するシーケンスデータ作成手段を更に備えた請求項１～３のいずれかに記載の電子楽器の音素材処理装置。

【請求項６】該該シーケンスデータに基づいて各子波形を再生する再生手段と、

該再生手段による再生の際に該シーケンスデータのタイミング情報を他の演奏データのタイミング情報に合わせるよう調節することで他の演奏データとの同期をとる同期手段とを更に備えた請求項５記載の電子楽器の音素材処理装置。

2

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は例えば演奏者が行った生演奏などの音素材を一音一音に分割する電子楽器の音素材処理装置に関するものである。

【０００２】

【従来の技術】生演奏の例えばリズム楽器音や音声、種々の効果音などの音素材を予め録音しておいて、これを他の演奏音と同期させて演奏する手法がある。例えば特開平７－２４４４８０号公報には、演奏者が生演奏した一連の演奏音をＰＣＭ録音し、更にその録音した演奏データを時間軸上で例えば等容量毎に複数のデータに分割してブロック化するとともに各ブロック化された演奏データにキーナンバー（ノート番号）を割り当てておき、他の演奏音の演奏タイミングに合わせて人がキーボード等により押鍵してそのキーナンバーに対応したブロックの演奏データを読み出し演奏させることで、他の演奏音との同期多重録音を容易に行えるようにした電子楽器の演奏データ処理方法について記載されている。

【０００３】

【発明が解決しようとする課題】しかし、上述の処理方法では、演奏データを分割してブロック化する際に、等容量毎に複数に分割するものであるので、分割されたブロックが演奏のビートの細かさと対応しなくなるおそれがある。次表はこれを説明するもので、１６ビートと４ビートの１小節分の演奏を等容量毎に分割した際の分割の態様を示すものである。

【０００４】

〔表：１小節の演奏を等容量毎分割〕

分割数→	４	１６
１６ビートの演奏	１つのブロックに４つの演奏音	１つのブロックに１つの演奏音
４ビートの演奏	１つのブロックに１つの演奏音	１つのブロックに１／４の演奏音

【０００５】例えば、図１１（ａ）は１６ビートの演奏データを４ブロックに分割したもので１つのブロックには４つの演奏音が入っている。この４ブロックに分割した１６ビートの演奏データを演奏速度が半分の他の演奏音と同期させた場合には、図１１（ｂ）に示すように、他の演奏音のタイミングに合わせて押鍵する毎に対応するキーナンバーのブロックの４つの演奏音が元の演奏データと同じテンポで演奏されるが、その後は次の押鍵があるまで空白期間となるため、不自然な演奏音になる。

【０００６】なお、この１６ビートの演奏データを１６ブロックに分割した場合には１つのブロックに１つの演奏音が入るため、他の演奏音のタイミングに合わせて押鍵する毎に１つの演奏音が演奏されることになるので、問題はない。

【０００７】また、図１１（ｃ）は４ビートの演奏データを４ブロックに分割したもので、１つのブロックには

１つの演奏音が入っている。この場合には問題ない。しかし、図１１（ｄ）に示されるように、この４ビートの演奏データを１６ブロックに分割した場合には、１つのブロックに１つの演奏音波形の１／４部分が入ることになるので、これを再生すると不自然な演奏音となる。

【０００８】この対策として、ブロックに分割する際に分割するブロック数を選択できるように構成することも考えられるが、この方法でも、演奏の一部のみが細かなビートで演奏されている場合などには対応できない。

【０００９】したがって、本発明は、生演奏などから得た音素材を、演奏のビートの細かさにも対応して音単位に的確に分割しブロック化できるようにすることを目的とする。また、分割結果の個々のデータを即座に自動演奏できるようにすることも目的とする。

【００１０】

【課題を解決するための手段】上述の課題を解決するために、本発明に係る電子楽器の音素材処理装置は、複数の一連の音からなる音素材をサンプリングした親波形を記憶する第１の記憶手段と、該第１の記憶手段から読み出した親波形を閾値を用いて該音の立上りを検出して複数の子波形に分割する分割手段と、該分割手段による分割結果を記憶する第２の記憶手段とを備える。このように閾値を用いることで、音素材を演奏のビートの細かさに対応して的確に分割、ブロック化することが容易にできる。

【００１１】上述の分割手段は、分割済の１つの子波形中に２以上の音が含まれているときに、該子波形について、該子波形を分割するに用いた閾値と異なる閾値を用いて該音の立上りを更に検出し直して該子波形を更に複数の子波形に分割する手段を含むように構成してもよい。

【００１２】あるいは上述の分割手段は、分割済の１つの子波形中に２以上の音が含まれているときに、該子波形について、該子波形の時間軸上で時間位置を指定して該子波形を更に複数の子波形に分割する手段を含むように構成してもよい。

【００１３】また本発明に係る電子楽器の音素材処理装置は、該分割した各子波形を指定する指定手段と、該指定手段で指定された子波形を再生する再生手段を更に備える。このような指定手段で各子波形を順次に指定することで元の親波形を再生することもできる。

【００１４】また本発明に係る電子楽器の音素材処理装置は、該第２の記憶手段に記憶された分割結果の子波形データに基づいて各子波形を一連に発音するタイミング情報を含んだシーケンスデータを作成するシーケンスデータ作成手段を更に備える。このシーケンスデータに基づいて、分割した個々の子波形を即座に自動演奏することができ、元の親波形を再生することもできる。

【００１５】また本発明に係る電子楽器の音素材処理装置は、このシーケンスデータ再生手段を備えたものにおいて、シーケンスデータに基づいて各子波形を再生する再生手段と、再生手段による再生の際に該シーケンスデータのタイミング情報を他の演奏データのタイミング情報に合わせるよう調節することで他の演奏データとの同期をとる同期手段とを更に備える。このような同期手段を用いることで、他の演奏データとの同期を容易にとることができる。

【００１６】

【発明の実施の形態】以下、図面を参照して本発明の実施形態を説明する。図２は本発明の一実施例としての電子楽器を示すブロック図であり、この電子楽器は演奏データを分割し、テンポ情報に同期させるための機能を備えている。この電子楽器システムは、演奏データ分割処理と全体の制御を行う中央処理装置（ＣＰＵ）１、装置全体の制御のためのプログラム等が格納されたリード・オンリー・メモリ（ＲＯＭ）２、演奏データが格納されたりＣＰＵのワーキングエリアとして使用されるランダム・アクセス・メモリ（ＲＡＭ）３、制御パラメータを確認しながら操作を行う表示部４と操作子部５、演奏データの入力を行うＡ／Ｄ変換部６、演奏データの出力を行うＤ／Ａ変換部７、出力される演奏データの音色等を制御する音色制御部８、演奏データやＭＩＤＩクロックデータの入出力を行うＭＩＤＩ入出力端子９を含み構成される。

【００１７】図３にはこの電子楽器の操作を行う操作パネルの例が示される。この操作パネルは制御パラメータ等を表示する表示部４とパラメータの設定等の各種操作を行う操作子部５からなる。操作子部５は５１～５５の各種操作子からなる。ここで、５１は表示部４に表示されたパラメータの値を変化させて設定するためのダイヤル式のロータリ・エンコーダであり、このロータリ・エンコーダ５１で設定されるパラメータは、閾値（スレッショルドレベル）、開始ポイント、終了ポイント、ピッチ、レベルがある。５２はＹＥＳ／ＮＯの指示を選択するスイッチ、５３は演奏データの分割処理を指示するための解析（アナライズ）スイッチ、５４は表示部４に表示されるパラメータを選択するためのパラメータ選択スイッチ、５５$_1$～５５$_4$は分割した演奏データを発音させ確認するための再生用スイッチである。なお、図示の例ではこの再生用スイッチは４つのみを示しているが、実際は分割する子ウェーブの数分だけ用意する。

【００１８】ここで本明細書では、分割する前の一連の演奏データを親ウェーブ、分割された個々の演奏データを子ウェーブと呼ぶことにする。再生用スイッチ５５$_1$～５５$_4$は、後述するように、分割された複数の子ウェーブにそれぞれ対応付けられており、この再生用スイッチ５５$_1$～５５$_4$の何れかを押すことでそれに対応する分割された子ウェーブが再生される。また、分割処理の対象となる親ウェーブが複数ある場合にそれらの親ウェーブにもそれぞれ対応付けられており、解析スイッチ５

３を押しながら再生用スイッチ５５$_1$～５５$_4$のどれかを押した場合には、その押した再生用スイッチに対応した親ウェーブについて演奏データ再構築モードに入るようになっている。

【００１９】この実施例の電子楽器の動作を説明する。まず、原理的な動作について図１を参照して説明する。

【００２０】図１は、ある演奏データを分割した場合の例を示すもので、図１（ａ）は元の親ウェーブの波形を示し、図１（ｂ）は分割処理の態様を示すものである。図中の波形を形成している各棒線は演奏音波形をサンプリングした各サンプル値を表す。図１（ｂ）に示すように、親ウェーブに対し、第１回目の閾値ＴＨ１を設定し親ウェーブがこの閾値ＴＨ１を超えてから一旦閾値ＴＨ１以下に減少し再び超えるまでの部分を１つのブロックとして分割を行うと、７つの子ウェーブに分割される。

【００２１】この場合、１、２、３、４、６、７番目の子ウェーブについては、１つのブロック内における演奏音のピーク値は１つであるからそれぞれ１つの演奏音を含む子ウェーブといえる。しかし、５番目の子ウェーブについては、１つのブロック内に２つのピーク値があるから、この子ウェーブは２つの演奏音を含んでいるものである。よって、５番目の子ウェーブはさらに分割する必要があるので、５番目の子ウェーブを指定し、再度、閾値ＴＨ１よりも大きな適当な閾値ＴＨ２を設定して２回目の分割にリトライする。図１（ｂ）の例では、この２回目の分割により、第１回目の５番目の子ウェーブがさらに２つに分割され、その結果、２回の分割実行で８つの子ウェーブに分割された状態が示されている。

【００２２】なお、ノイズ的なピーク値に対してもブロック化されてしまわないように、１ブロックの最小時間（例えば１００ｍｓ）をあらかじめ決めておいて、それより短いブロックが作成されそうな場合は、その後のブロックに自動的に組み込まれるようにしてもよい。

【００２３】上述の原理的な動作を行うための処理を以下に説明する。図４は演奏データ再構築ルーチンのフローチャートであり、この演奏データ再構築ルーチンは、図示しないメインルーチン中のパネル処理ルーチン（パネル上のスイッチなどをスキャンして状態を調べるルーチン）において、解析スイッチ５３を押しながら再生用スイッチ５５$_1$～５５$_4$のどれかを押したのを検出したことを契機として読み出されて、その押した再生用スイッチに対応した親ウェーブについて演奏データ再構築処理を行う。

【００２４】演奏データ再構築モードに入ると、該当する親ウェーブが既に分割済であるか調べ（ステップＳ２）、既に分割済である場合は、所定のスイッチを押すことによって、既に分割されている演奏データを再生するプレイモードか、さらに分割し直す解析モード（アナライズモード）かを選択する（ステップＳ３）。プレイモードの場合にはこの演奏データ再構築ルーチンを終了する（ステップＳ１０）。ステップＳ２で親ウェーブがまだ分割済でないと判定された場合は解析モードになる。

【００２５】解析モードにおいては、該当する親ウェーブを形成しているサンプル値のうちの最大値と最小値を検索し（ステップＳ４）、検索された最大値から最小値を引いた値を１２８分割したものを分解能の単位とし、この分解能で操作子を用いて閾値（センス値）の設定を行う（ステップＳ５）。この設定は操作者が表示部４に表示された閾値を見ながら手動で調整してもよいし、１回目、２回目の分割処理に対しあらかじめ定められた値を制御プログラムが自動的に設定するものであってもよい。なお、閾値の設定にあたっては、親ウェーブの絶対値の波形をディレイに表示してユーザが閾値を設定しやすいようにする。

【００２６】閾値の設定が終わると、表示部４に分割を行うか否かの問い合わせ表示がされるので、分割を行う場合はＹＥＳスイッチ５２を押し、分割しない場合はＮＯスイッチ５２を押す（ステップＳ６）。ＹＥＳスイッチ５２を押すと分割処理が実行され（ステップＳ７）、ＮＯスイッチ５２を押すとこの演奏データ再構築ルーチンを終了する（ステップＳ１０）。この分割処理（ステップＳ７）の詳細は図５に示すフローチャートを参照して後述する。

【００２７】分割が終了すると、分割された演奏データと、その発音タイミングに基づき、シーケンスデータが作成される（ステップＳ８）。このシーケンスデータの作成方法については後述する。

【００２８】分割処理が終了すると、表示部４に分割処理を再実行するかの問い合わせ表示がされる（ステップＳ９）。このとき、再生用スイッチ５５$_1$～５５$_4$には今行った分割処理で分割された個々の演奏データ（子ウェーブ）を演奏するシーケンスデータが割り当てられており、この再生用スイッチ５５$_1$～５５$_4$を操作することによって分割された個々の演奏データをそれぞれ再生して確認することができる。この再生確認によって、親ウェーブの分割が正しく行われたか否かを耳で聴いて判断できるので、正しくできている場合には再実行“ＮＯ”をＮＯスイッチ５２で指示する。正しくできていない場合には再実行“ＹＥＳ”をＹＥＳスイッチ５２で指示する。この際、表示部４の表示を見ながら操作子部５を操作することによって、親ウェーブの分割を閾値を設定し直してもう一度最初から全部やり直すか（ＡＬＬ選択）、あるいは分割が正常にできなかった一部の子ウェーブについてだけ更に閾値を再設定して分割処理をするか、を選択することができる。

【００２９】再実行“ＹＥＳ”の指示がされたときはステップＳ５～Ｓ８を繰り返す。一部の子ウェーブについてだけ更に閾値を設定して分割処理をする場合は、これが２回目以降の分割処理となる。このような分割処理を

繰り返すことによって、最終的に、全ての子ウェーブが１つの演奏音だけからなるように、親ウェーブを分割することができる。

【００３０】次に図５のフローチャートを参照して演奏データの分割処理について説明する。分割の対象となる親ウェーブの演奏データの先頭にポインタＰＴＲを設定する（ステップＳ７０２）。このポインタＰＴＲは以降、逐次に１つずつインクリメントされてウェーブの処理位置をサンプリング数の単位で示す。なお、このポインタＰＴＲの設定では、図４のフローチャートの再実行ステップ９において再実行が指示された場合、その際にＡＬＬ（親ウェーブの分割処理のやり直し）が選択されていれば親ウェーブの先頭が、また任意の子ウェーブが選択されていればその子ウェーブの先頭が設定される。

【００３１】次いで、先頭の子ウェーブ用パラメータに開始ポイント（子ウェーブのブロックの先頭位置）を設定する。すなわち子ウェーブの開始ポイントを記憶しておくパラメータバッファに開始ポイントを書き込む（ステップＳ７０３）。この場合も、再実行ステップＳ９においてＡＬＬが選択されているときは先頭の子ウェーブのパラメータバッファに親ウェーブの先頭を開始ポイントとして設定し、任意の子ウェーブが選択されている場合にはその子ウェーブのパラメータバッファに書き込まれている開始ポイントをそのまま用いる。

【００３２】なお、この子ウェーブのパラメータバッファには後述する終了ポイント（子ウェーブのブロックの最後尾位置）も書き込まれる。

【００３３】次いで、フラグＦＬＡＧをオフにする（ステップＳ７０４）。このフラグＦＬＡＧは、次の子ウェーブの先頭を認識するためのもので、子ウェーブのサンプル値が閾値よりも小さくなったこと（サンプル値＜閾値）でオンされ、次にそのオン状態で閾値よりも大きいサンプル値（サンプル値≧閾値）を検出したら、それが次の子ウェーブの先頭であることを示す。

【００３４】なお、上記ではサンプル値が閾値より小さいとフラグを反転とすると説明したが、より具体的には、ゼロクロス点付近で頻繁にフラグが反転してしまうことを防ぐために、所定回数サンプル値が閾値より小さいことを確認してから初めてフラグをオンにする。あるいは、他の実施例として、例えば１０ｍｓ毎の絶対値の最大値を順次に閾値と比較する。図１、図１０の各振幅はこの時間毎の絶対値の最大値を表示したものであって、この最大値に基づいて分割点を決定する。

【００３５】以降、ポインタＰＴＲから演奏データの振幅レベル（サンプル値）を１サンプル毎に検索していき、サンプル値＞閾値か否かを判定する（ステップＳ７０５）。サンプル値が閾値よりも大きい場合、すなわち演奏音が検出された場合には（ステップＳ７０５）、さらにフラグＦＬＡＧがオンか否かを判定し（ステップＳ７０６）、フラグＦＬＡＧがオフの場合は、ポインタを１つ更新して（ステップＳ７０７）、次のサンプル値と比較する（ステップＳ７０５）。

【００３６】サンプル値が閾値よりも小さくなった場合、すなわち検出された演奏音が終了したと判定された場合は、ポインタＰＴＲを１つ更新するとともにフラグＦＬＡＧをオンして（ステップＳ７０８）、次のサンプル値との比較を行う（ステップＳ７０５）。

【００３７】上記のステップＳ７０６でフラグＦＬＡＧがオンと判定される状態は、演奏音が終了したと判定されてから再び演奏音が検出されたことを意味する。すなわち、次の子ウェーブのブロックの先頭が検出されたことになる。この場合には、子ウェーブの長さが一定以上に達しているか否かを判定する（ステップＳ７０９）。一定の長さ以上に達していない場合は、上記演奏音と思って検出したものがノイズである可能性が高いので、フラグＦＬＡＧを再びオフにして（ステップＳ７１０）、ポインタを更新し（ステップＳ７０７）、次のサンプル値との比較を再び行う（ステップＳ７０５）。このようにステップＳ７０９で子ウェーブの長さが一定以上か見るのは、上述のようにノイズ除去のためであり、また、子ウェーブの番号が不連続にならないよう番号の付け替えも行う。

【００３８】子ウェーブの長さが一定以上に達している場合は（ステップＳ７０９）、子ウェーブの終了ポイントを記憶しておくパラメータバッファに終了ポイントを書き込むとともに、フラグＦＬＡＧをオフに設定する（ステップＳ７１１）。これによって、子ウェーブのパラメータバッファには、分割処理がされた結果として、親ウェーブ中の任意のポイントが、子ウェーブの開始ポイントと終了ポイントとして書き込まれる。なお、このようにフラグがオフになったところを分割点とするものであるが、より詳しく説明すると、分割点と判定したすぐ前の（マイナスからプラスへの）ゼロクロス点で最初にプラスとなったアドレスを開始ポイントとし、その一つ前のアドレスを終了アドレスとする。

【００３９】なお、この実施例では最終アドレスと開始アドレスをこの時点で検出しているが、子ウェースの最終アドレスは、各サンプルのレベルが閾値より所定回数小さくなってから次のゼロクロス点を最終アドレスとし、次の子アドレスの開始アドレスは上述のように閾値レベルを超えた直前のゼロクロスアドレスとするようにしてもよい。また、最終アドレスを検出するための閾値レベルと開始アドレスを検出するための閾値レベルをそれぞれ設定できるようにしてもよい。

【００４０】次いでポインタＰＴＲを１つ更新して（ステップＳ７１２）、次のサンプル値を参照し、分割の対象となるウェーブの終端に達しているか否かを判定する（ステップＳ７１３）。ウェーブの終端でなければ、次の子ウェーブの開始ポイントを記憶しておくパラメータバッファに開始ポイントを書き込み（ステップＳ７１

４）、ステップＳ７０５以降の処理を繰り返す。ウェーブの終端であれば、この分割処理ルーチンを終了する（ステップＳ７１５）。

【００４１】次に、上記の分割処理によって得た複数の子ウェーブに基づき親ウェーブのシーケンスデータを作成する方法について図７、図８を参照して説明する。図７に示されるように、上記の分割処理で得た複数の子ウェーブとそのパラメータに基づき、各子ウェーブに先頭から順番にウェーブ番号を付けるとともに、分割された各子ウェーブの開始ポイントを基にして、各子ウェーブの発音タイミングからゲートタイムを計算し、シーケンスデータを作成する。このゲートタイムは一の子ウェーブの発音時から次の子ウェーブ発音までのインターバルである。シーケンスデータの構成は、〔ゲートタイム＋子ウェーブ番号〕になっていて、ゲートタイム経過後に子ウェーブを開始ポイントから終了ポイントまで再生する。このゲートタイムと子ウェーブ番号のデータはそれぞれ１バイトからなる。

【００４２】このシーケンスデータの作成手順は次のとおりである。

①最初に再生される子ウェーブ（表では第１番）の開始ポイントをそのまま先頭のゲートタイムとする。

②２番目に再生される子ウェーブの開始ポイントから、最初に再生される子ウェーブの開始ポイントを引いた値を、２番目のゲートタイムとする。

③以下、Ｎ＋１番目に再生される子ウェーブの開始ポイントから、Ｎ番目に再生される子ウェーブの開始ポイントを引いた値をＮ番目のゲートタイムとする。

④最後のウェーブ番号は“ＦＦ”とする。再生時にはこの“ＦＦ”番号を再生するところで、再生を終了させる。

【００４３】図７にはシーケンスデータ作成のための子ウェーブのパラメータ例が、図８には作成されたシーケンスデータの例が示される。開始／終了ポイントおよびゲートタイムは１６進数で表示されている。ゲートタイムが“ＦＦ”の長さを超える場合には次のバイトもゲートタイムに用いられる。図８の例では４番目のゲートタイムは“ＦＦ＋１３”の長さである。演奏テンポの変更があった場合にはこのゲートタイムの値を変更する。

【００４４】また上述の子ウェーブ番号すなわち分割した個々の子ウェーブをそれぞれ再生用スイッチ55_1～55_4に割り当てる。

【００４５】上記の子ウェーブ番号にノート番号（すなわちキーナンバー）を割り当ててもよい。図９はその例を示す。ここでは４種類の親ウェーブＡ、Ｂ、Ｃ、Ｄがあり、各親ウェーブＡ、Ｂ、Ｃ、Ｄはそれぞれ１６個の子ウェーブＡ－１～Ａ１６、Ｂ－１～Ｂ１６、Ｃ－１～Ｃ１６、Ｄ－１～Ｄ１６に分割されている。子ウェーブＡ－１から順番に自動的に図示のようにノート番号を割り当てる。これにより、〔ゲートタイム＋ノート番号〕でシーケンスデータを作成することができ、シーケンスデータ中のノート番号を音階の順番で配置することで、シーケンスデータにより元の親ウェーブの演奏音を元のまま再生することができる。

【００４６】上記のシーケンスデータ例では、シーケンスデータのタイミング情報をゲートタイムにより表現し演奏テンポに合わせてこのゲートタイムを変えるするようにしたが、この他に、このタイミング情報をＭＩＤＩクロックで表現し外部等からのＭＩＤＩクロックを用いて演奏テンポに合わせるようにするものであってもよい。

【００４７】例えば、親ウェーブのサンプリング周波数をF_s〔Ｈｚ〕とすると、１サンプル当たりの所要時間は、

１サンプル＝１／F_s〔sec〕

である。ここで、親ウェーブのテンポをＴ〔bpm〕とする。この単位〔bpm〕は１分間当りの４分音符の数である。ＭＩＤＩクロックの長さT_{clk}は、４分音符１つが２４ＭＩＤＩクロックに相当するから、

T_{clk}＝Ｔ／６０×２４〔sec〕

である。よって、第１番目の子ウェーブから第２番目の子ウェーブが発音されるまでの時間は、N_1、N_2をそれぞれ第１、２番目の子ウェーブの開始ポイントとすると、

（N_2－N_1）／F_s〔sec〕＝〔（N_2－N_1）／F_s〕／〔Ｔ／（６０×２４）〕

単位〔ＭＩＤＩクロック〕

となる。

【００４８】このように、シーケンスデータのタイミング情報をＭＩＤＩクロックで表現すると、内部や外部からのＭＩＤＩクロックによって、上記シーケンスデータ中のタイミング情報を調整することで、上記シーケンスデータによる親ウェーブの再生音を他の演奏データの再生音に簡単に同期させることができる。

【００４９】次に、上記で作成したシーケンスデータの再生処理を説明する。図６はシーケンスデータ再生ルーチンのフローチャートである。この再生ルーチンは、図示しないメインルーチン中のパネル処理ルーチンで、再生用スイッチ55_1～55_4の何れかが押されたことによって読み出されて実行される（ステップＳ１１）。

【００５０】まず、シーケンスデータの先頭の１バイト（つまりゲートタイム）をカウンタエリアに格納する（ステップＳ１２）。この格納した値が“ＦＦ”であれば、フラグをオンにする。。このカウンタエリアの値は所定時間毎に逐次にデクリメントされることでカウンタとしての役目をする。またポインタをシーケンスデータの先頭に置く。このポインタはシーケンスデータ再生の進行をシーケンスデータのバイト単位で管理するためのものである。

【００５１】次に、タイマを起動する（ステップＳ１

３）。このタイマにより所定時間が経過する毎にタイマ割込みを発生させる。タイマ割込みがあるか否かを判定し（ステップＳ１４）、タイマ割込みを受けたら、カウンタエリアに設定した値（ゲートタイム）を１つカウントダウンする（ステップＳ１５）。カウンタの値がカウンタ値≦１か否かを判定し、「０」になっていなければタイマ割込み毎にカウンタの値のデクリメントを続ける（ステップＳ１４～Ｓ１６）。

【００５２】カウンタの値が「０」になったら（ステップＳ１７）、設定したゲートタイムが経過したことを意味するので、ポインタを１つカウントアップし（ステップＳ１７）、先にカウンタエリアに設定したデータ（ゲートタイム）が“ＦＦ”か否かをフラグを参照して判定する（ステップＳ１８）。このデータが“ＦＦ”であれば（ステップＳ７１４）、先にシーケンスデータ作成の説明で述べたようにゲータタイムは“ＦＦ＋○○”であり、まだゲートタイムが終了しておらずカウントダウンが続くので、シーケンスデータから“ＦＦ”の次のバイトのデータ（残りのゲートタイム○○）を読み出してカウンタに格納し（ステップＳ１９）、この値が「０」となるまでカウントダウンを繰り返す（ステップＳ１４～Ｓ１９）。

【００５３】ステップＳ１８の判定において、次のデータが“ＦＦ”でなければ、フラグをオフにリセットし、子ウェーブ番号をワークエリアに格納してから、カウンタにシーケンスデータから読み出した次のゲートタイムのデータをセットする（ステップＳ２０）。

【００５４】この後、子ウェーブの再生処理を行う（ステップＳ２１）。子ウェーブの再生処理が終わったら、シーケンスデータが終了か否かを判定し（ステップＳ２２）、この処理をシーケンスデータが終了するまで繰り返す（ステップＳ１４～Ｓ２２）。シーケンスデータが終了したら、このシーケンスデータ再生ルーチンを終了する（ステップＳ２３）。

【００５５】子ウェーブの再生にあたっては、音色制御部８によって、分割した子ウェーブの読出し毎に音色の制御を行って分割後の演奏音毎に音色の制御を行うようにしてもよい。また、この音色制御は、外部から入力された音色制御情報に対応して、読み出された子ウェーブの音色を制御するように構成してもよい。また、作成したシーケンスデータに音色等の制御情報を記録するようにし、音色制御部で分割した演奏データ毎に音色等の制御を行うようにしてもよい。

【００５６】なお、上述の実施例では、テンポが速くなると、前後の子ウェーブが重なって鳴ることになるので、この場合には、後発優先で先に鳴っていた子ウェーブの音を消す、あるいは重なる部分をクロスフェードさせるなどするとよい。

【００５７】本発明の実施にあたっては種々の変形形態が可能である。例えば、上述の実施例では、１回目の分割処理の結果、２つ以上の演奏音が含まれた子ウェーブが生じた場合には、２回目の分割処理を閾値を変えて行うことでこの子ウェーブをさらに細かく分割するようにしたが、本発明はこれに限られるものではなく、操作者がその２つ以上の演奏音を含む子ウェーブを再生し耳で聴いて子ウェーブ中のどの時間位置に各演奏音があるかを掴み、子ウェーブの２回目の分割位置を手動操作により子ウェーブの時間軸上で時間をパラメータにして指定することで、子ウェーブをさらに細かく分割するようにしてもよい。

【００５８】また、上述の実施例では、１回目の分割の後に、１音だけに分割し切れていないブロックを閾値を変えて２回目以降の分割を行うことで、１音だけのブロックに細分化しているが、これに代えて、初めから複数の閾値を用意しておいて、親ウェーブをこれらの複数の閾値と比較してそれぞれの比較結果を蓄えておいて、親ウェーブとの比較処理が終わった時点でこれらの比較結果に基づいて一挙に１音だけのブロックに分割していくのもであってもよい。

【００５９】また、上述の実施例では、親ウェーブのシーケンスデータを他の演奏データに同期させる方法として、ゲートタイムを変える方法とタイミング情報をＭＩＤＩクロックで表現してＭＩＤＩクロックで制御する方法を述べたが、本発明はこれに限られるものではなく、例えばシーケンスデータはそのままにして相対的に再生タイミングを変えるようにしてもよい。すなわち、前述の図６に示すシーケンスデータ再生ルーチンにおいて、ステップＳ１３で起動するタイマを、タイマカウンタＴＣの値を任意に設定できるようにしてこのタイマカウンタＴＣを一定時間間隔で１ずつ減少させてＴＣ＝０になった時にタイマから割込みが発生するように構成する。これによりＴＣの設定値が小さければ割込みが頻繁にかかり、ＴＣの設定値が大きければ割込みがかかる周期が長くなる。よって、再生タイミングを内部的に変える場合にはこのタイマカウンタＴＣの値を内部で任意に設定し、再生タイミングを外部から変える場合にはこのタイマカウンタＴＣの値を外部クロックを用いて設定するようにする。

【００６０】また上述の実施例では、分割した演奏データ（子ウェーブ）を再生する場合には元の演奏データ（親ウェーブ）からその子ウェーブに対応する箇所を部分的に読み出すようにしたが、分割したことによって得られた開始ポイントと終了ポイントに対応する区間の演奏データを各子ウェーブの演奏データとして別途メモリに書き込むようにしてもよい。

【００６１】

【発明の効果】以上に説明したように、本発明によれば、生演奏などから得た音素材を、演奏のビートの細かさに対応して的確に分割、ブロック化することができる。また、分割結果から得たシーケンスデータを用いれ

ば、分割した個々の子波形演奏データを即座に自動演奏することができる。

【図面の簡単な説明】

【図１】本発明の実施例の原理的な動作を説明するために図である。

【図２】本発明の実施例装置のブロック構成を示す図である。

【図３】実施例装置の操作パネルの例を示す図である。

【図４】実施例装置において実行される演奏データ再構築ルーチンのフローチャートである。

【図５】実施例装置において実行される演奏データ分割処理ルーチンのフローチャートである。

【図６】実施例装置において実行されるシーケンスデータ再生ルーチンのフローチャートである。

【図７】実施例装置におけるシーケンスデータ作成のための子ウェーブ・パラメータの例を示す図である。

【図８】実施例装置におけるシーケンスデータの例を示す図である。

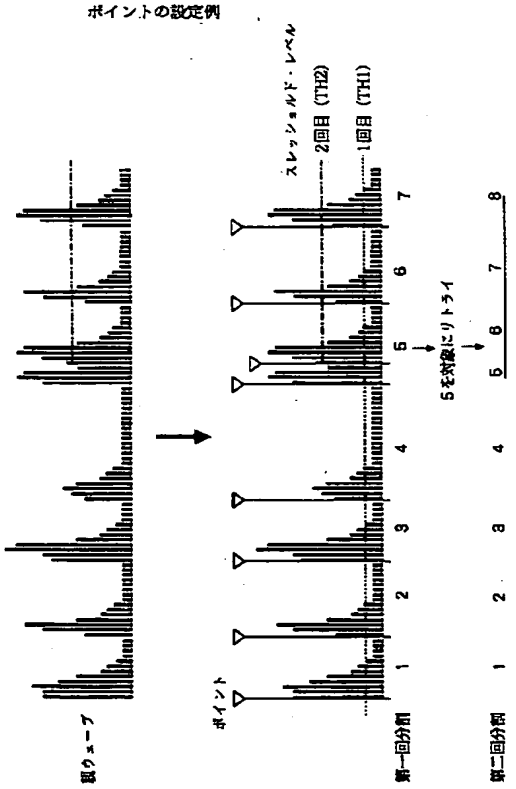
【図９】子ウェーブ番号にノート番号を割り当てる例を示す図である。

【図１０】ブロックの分割と演奏のビートの細かさの対応の問題点を説明する図である。

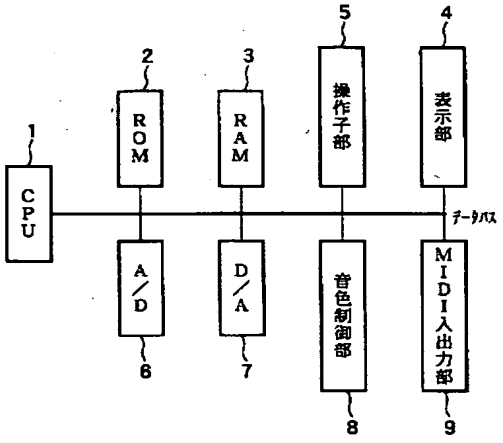
【符号の説明】

１　ＣＰＵ	２　ＲＯＭ
３　ＲＡＭ	４　表示部
５　操作子部	６　Ａ／Ｄ変換部
７　Ｄ／Ａ変換部	８　音色制御部
９　ＭＩＤＩ入出力部	
５１　ロータリエンコーダ	５２　ＹＥＳ／ＮＯスイッチ
５３　解析スイッチ	５４　パラメータ選択スイッチ
５５$_1$ ～５５$_4$　再生用スイッチ	

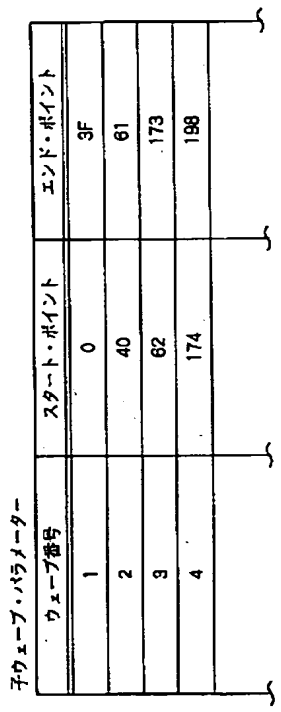
【図１】



【図２】

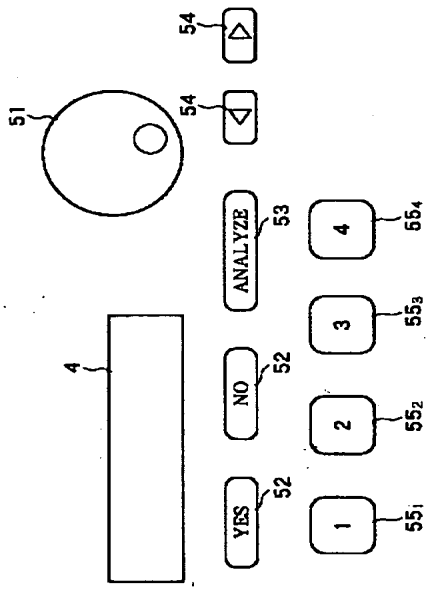


【図７】



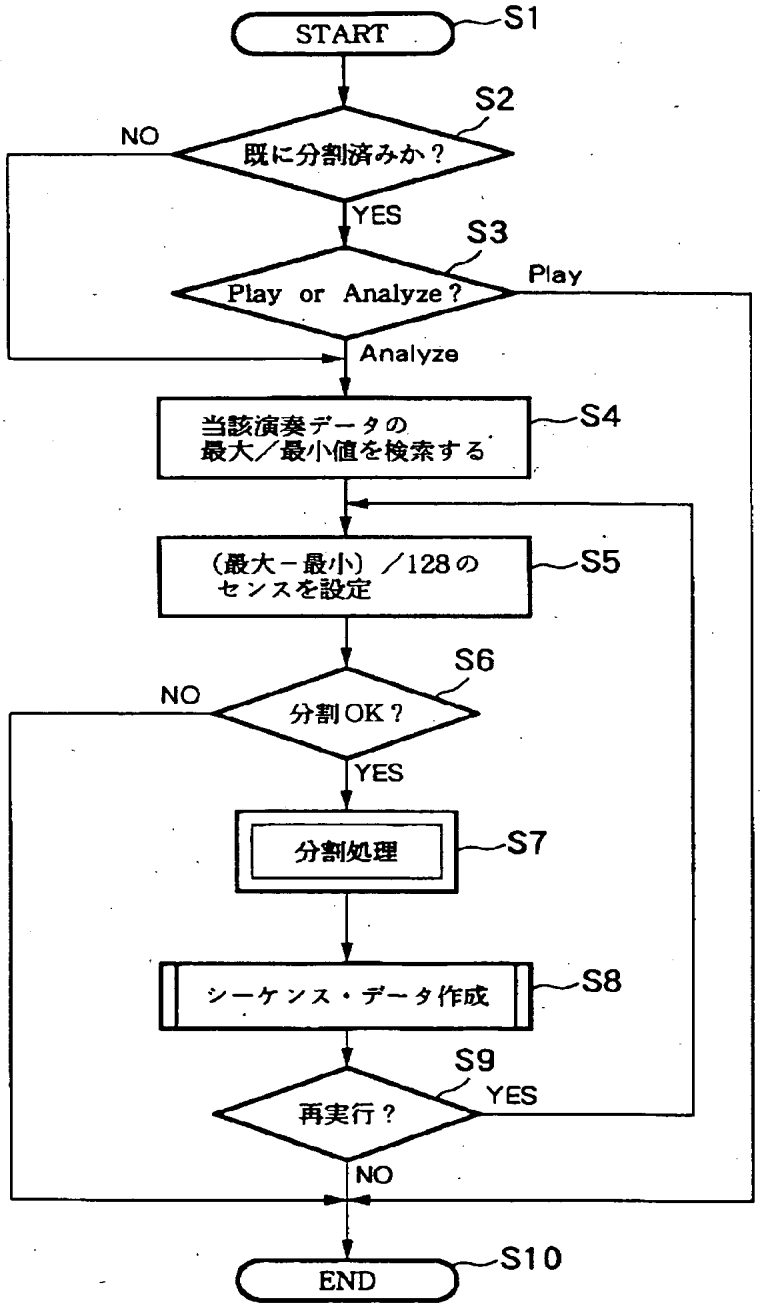
【図３】

操作パネル例



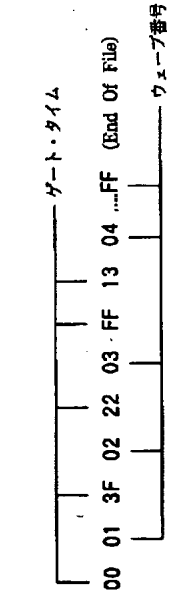
【図４】

演奏データ再構築処理

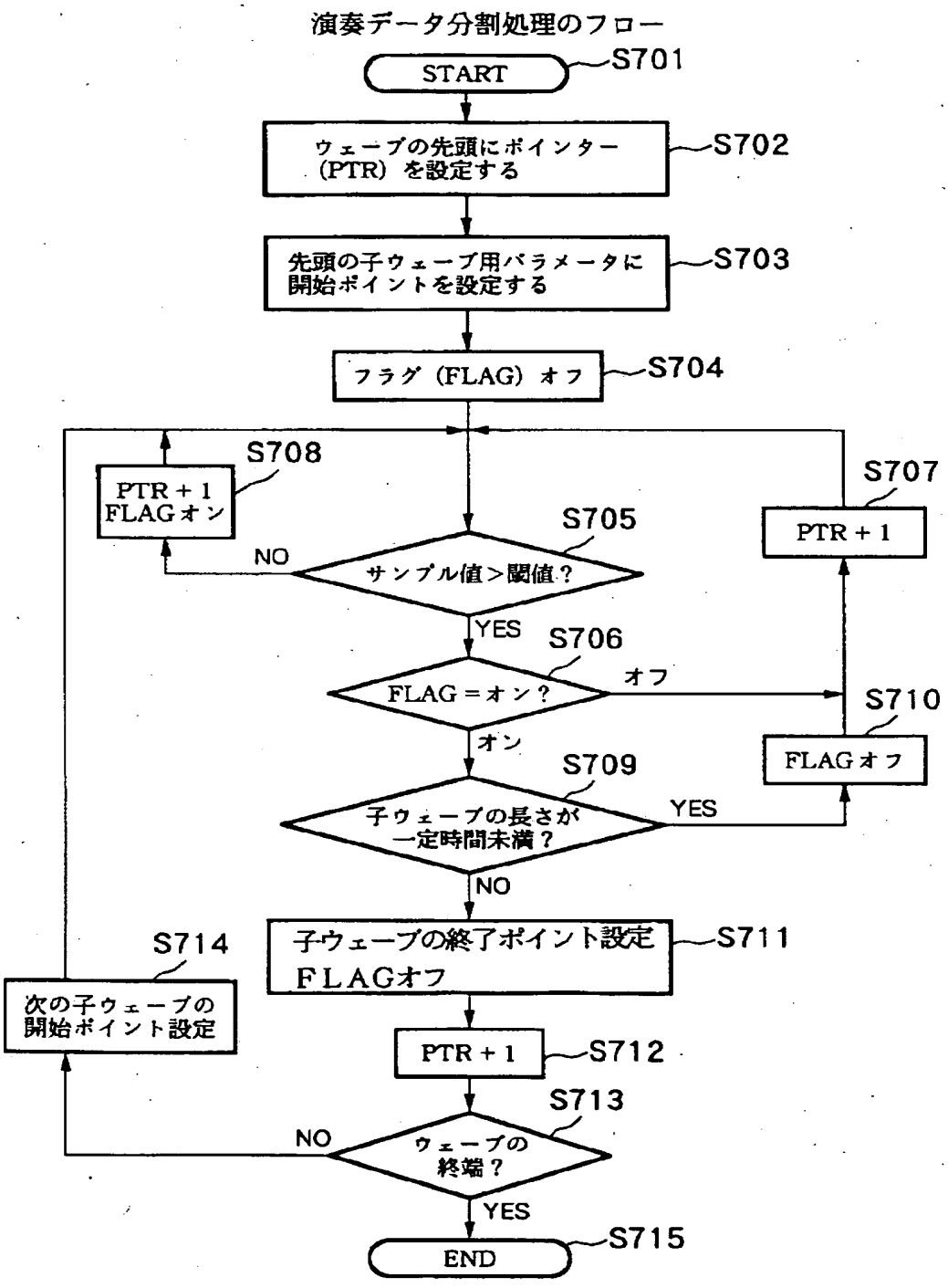


【図８】

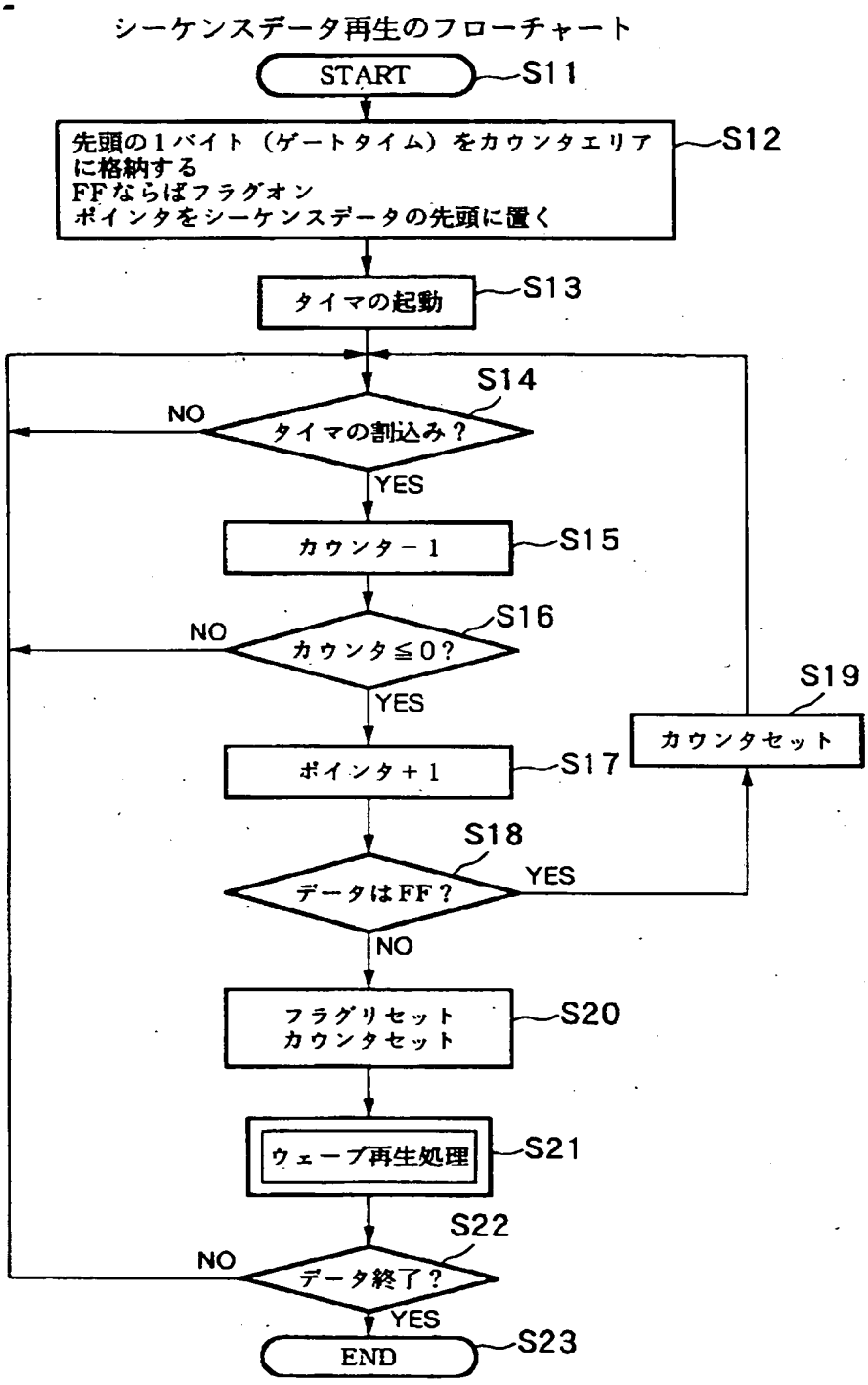
シーケンス・データ例



【図５】


演奏データ分割処理のフロー
START
S701
ウェーブの先頭にポインター（PTR）を設定する
S702
先頭の子ウェーブ用パラメータに開始ポイントを設定する
S703
フラグ（FLAG）オフ
S704
S705
サンプル値＞閾値？
NO
YES
S706
FLAG＝オン？
オフ
オン
S707
PTR＋1
S708
PTR＋1
FLAGオン
S709
子ウェーブの長さが一定時間未満？
YES
NO
S710
FLAGオフ
S711
子ウェーブの終了ポイント設定
FLAGオフ
S712
PTR＋1
S713
ウェーブの終端？
NO
YES
S714
次の子ウェーブの開始ポイント設定
S715
END

【図６】


シーケンスデータ再生のフローチャート
START
S11
先頭の１バイト（ゲートタイム）をカウンタエリアに格納する
FFならばフラグオン
ポインタをシーケンスデータの先頭に置く
S12
タイマの起動
S13
S14
NO
タイマの割込み？
YES
カウンタ－１
S15
S16
NO
カウンタ≦０？
YES
ポインタ＋１
S17
S18
データはFF？
YES
NO
S19
カウンタセット
フラグリセット
カウンタセット
S20
ウェーブ再生処理
S21
S22
NO
データ終了？
YES
END
S23

【図９】

分割した個々のウェーブへのノート番号の割り当て例

パッド・ノート番号（仮）対応表																
A-1 C1	A-2 C#1	A-3 D1	A-4 D#1	A-5 E1	A-6 F1	A-7 F#1	A-8 G1	A-9 G#1	A-10 A1	A-11 A#1	A-12 B1	A-13 C2	A-14 C#2	A-15 D2	A-16 D#2
B-1 C3	B-2 C#3	B-3 D3	B-4 D#3	B-5 E3	B-6 F3	B-7 F#3	B-8 G3	B-9 G#3	B-10 A3	B-11 A#3	B-12 B3	B-13 C4	B-14 C#4	B-15 D4	B-16 D#4
C-1 C5	C-2 C#5	C-3 D5	C-4 D#5	C-5 E5	C-6 F5	C-7 F#5	C-8 G5	C-9 G#5	C-10 A5	C-11 A#5	C-12 B5	C-13 C6	C-14 C#6	C-15 D6	C-16 D#6
D-1 C7	D-1 C#7	D-3 D7	D-4 D#7	D-5 E7	D-6 F7	D-7 F#7	D-8 G7	D-9 G#7	D-10 A7	D-11 A#7	D-12 B7	D-13 C8	D-14 C#8	D-15 D8	D-16 D#8

【図１０】